Operating Instructions

JETI VersaFlic Software and Firmware

Software V1.1.0 and newer

Firmware V1.6.1 and newer

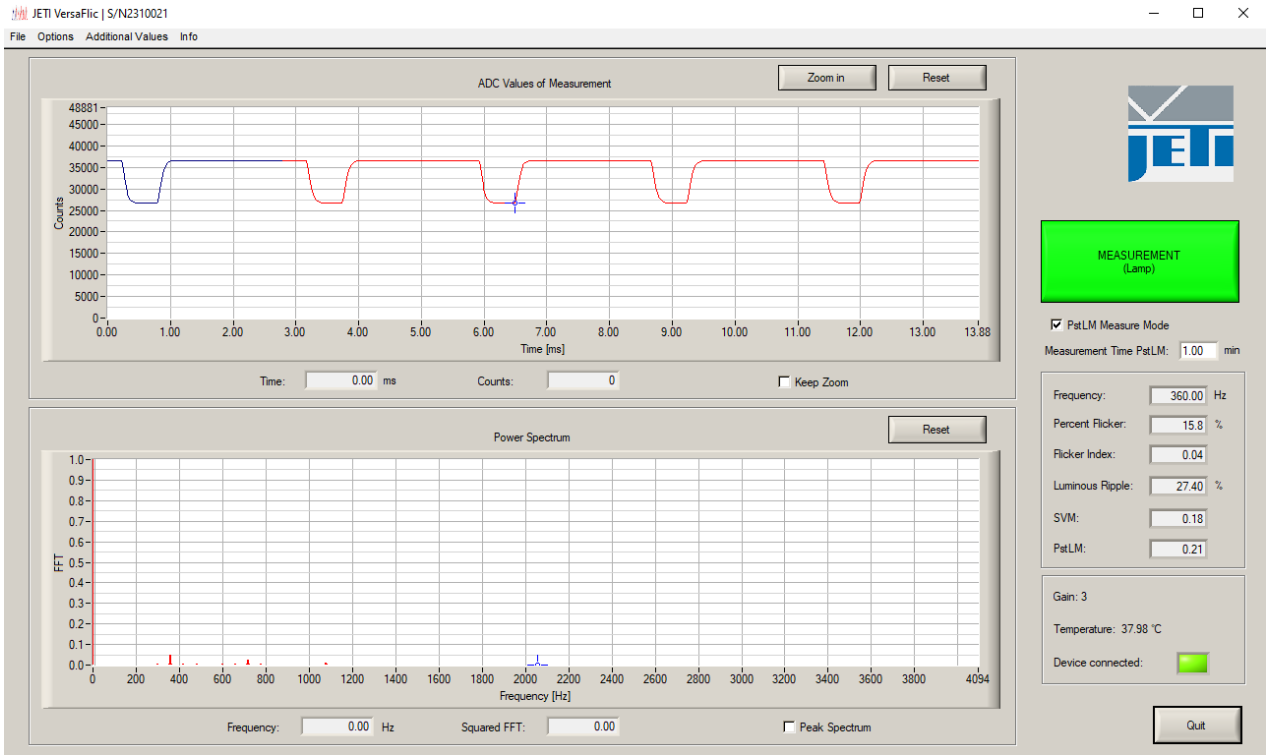
JETI Technische Instrumente GmbH
Göschwitzer Str. 48
D-07745 Jena
Tel.: +49-3641-23292 00
Fax: +49-3641-23292 01
E-mail: sales@jeti.com
Internet: www.jeti.com

Table of Contents

1 Software Documentation

1.1 What is JETI VersaFlic?

JETI VersaFlic is a flicker measurement software for the JETI ACC Flicker device. It comes in two variants, allowing measurement of flicker in lamps and displays. In the case of lamps, it calculates percentage flicker, flicker index, dominant frequency, luminous ripple, PstLM and SVM. In the case of displays, it measures flicker frequency, contrast and JEITA flicker values. In addition, it also shows recorded samples and power spectra.



The main Flicker Software menu

1.2 Installation

1.2.1 Hardware installation

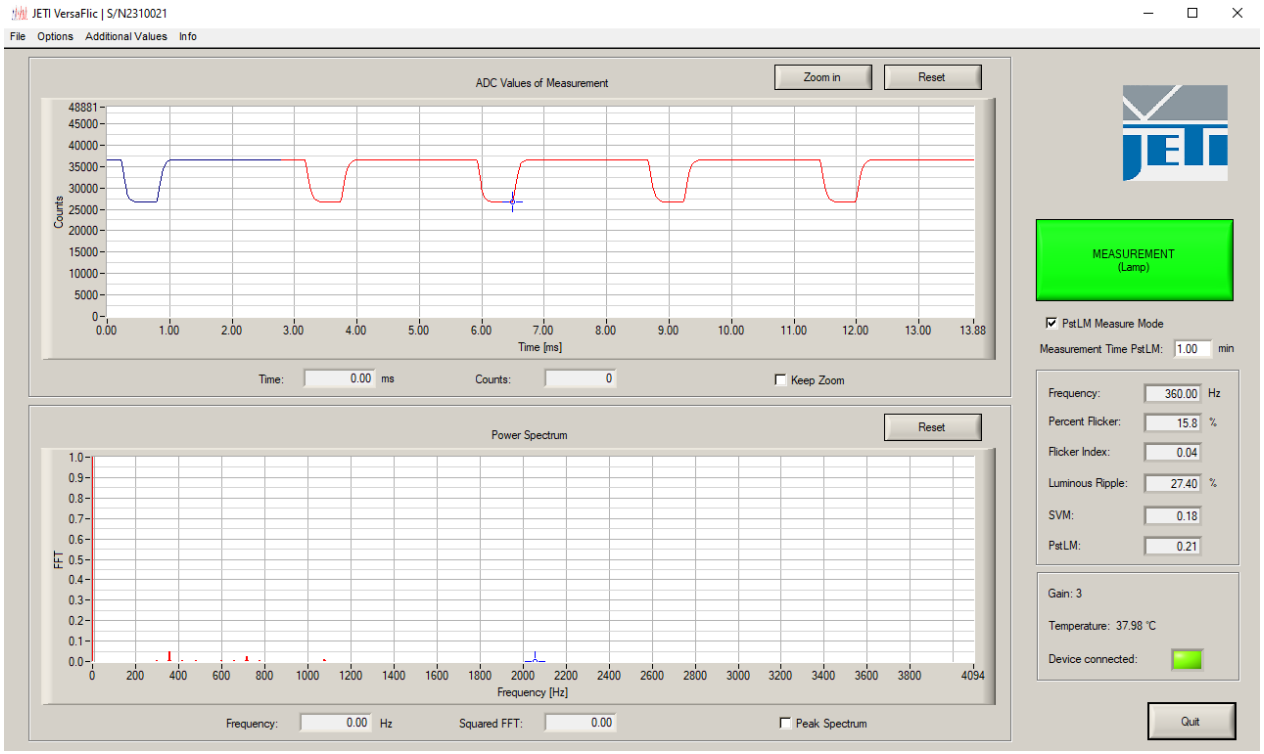
Simply connect the USB cable of the flicker device to a USB port. Run the accompanying software, which will automatically detect the Flicker measurement device, and start measurements.

1.2.2 Software installation

Navigate to the installer and click install.exe. The setup program will take you through the necessary steps. A shortcut will be created on the desktop.

1.3 Flicker Software

1.3.1 Performing a Measurement



The main Flicker Software menu

Measurement (Lamp) Button

Only available if a JETI ACC Flicker device for lamps is connected. Performs a flicker measurement and displays detected frequency, flicker percentage, flicker index, luminous ripple and SVM. In addition, sample data and power spectrum are shown in the graphs.

Measurement (Display) Button

Only available if a JETI ACC Flicker device for displays is connected. Performs a flicker measurement and displays detected frequency, flicker contrast and JEITA value. In addition, sample data and power spectrum are shown in the graphs.

Pst-LM Measurement Mode

The small tick box below the measurement button activates the Pst-LM (Short Term Perceptibility for light modulation) measurement mode. This is a much longer measurement than the standard percentage flicker or flicker index. The desired time can be chosen from 1 - 10 minutes.

If the tick box is active, a PstLM measurement is performed after the standard flicker measurement. An optional beep at the end of the measurement can be activated/deactivated choosing Options -> Beep at end of PstLM measurement.

1.3.2 Sample and Power Spectrum

The sample graph (the upper graph with title “ADC Values of Measurement”) displays the first five periods of a recorded sample, to easily identify the shape and frequency of flicker. The power spectrum graph displays the Fourier Transform with the major frequency components of the signal to easily compare their relative strength.

Zooming

Both graphs can be zoomed in by holding CTRL, clicking and drawing a box with the mouse. Clicking on the **Zoom in** button shows the flicker without starting at zero. The **Reset** button reverts the graph to its original.

Eye Sensitivity

This option is only available when using display measurement devices. It shows the power spectrum graph as weighed for calculating JEITA values. Frequencies larger than 60 Hz are strongly suppressed in this case.

Peak Spectrum

This option shows the spectrum with the DC peak suppressed.

1.3.3 Results

In the box below the measurement button, the results are shown. For displays, this is frequency, flicker contrast and JEITA value. For lamps, frequency, flicker percentage, index, luminous ripple and SVM are shown.

Hovering over the box with the results will display a tooltip, showing the formulas used to calculate the various values (With the exception of the more complicated SVM and PstLM).

1.3.4 Menu Bar

1.3.4.1 File

Connect/Disconnect

Should it be necessary to manually connect/disconnect the device – because it should be accessed by another software or for similar reasons – you can connect or disconnect the device by clicking on this option.

Save

Saves the calculated values (Frequency, Contrast Flicker and JEITA Value for display measurements, Frequency, Percent Flicker, Flicker Index, Luminous Ripple, SVM and optionally PstLM for lamp measurements), the raw sample data and a screenshot of the current sample and power spectrum graphs.

Quit

Quits the software.

1.3.4.2 Options

Automatic Gain

Switches measurement mode to automatic adaptation.

Manual Gain

Prompts for a fixed gain [1 - 3] that is used for the measurements. This could lead to overexposure and must be set accordingly.

Debug

Opens a debug popup. This is right now password protected and should only be used by advanced users. It allows to change offset and gain of the device, to perform a dark calibration and a debug measurement. It also allows to send custom commands and receive the data, which was sent by the device.

Dark Calibration: A dark/background noise measurement over a longer time and temperature range is performed. This can take up to half an hour. The determined parameters for dark and temperature correction must be saved with ‘Save Parameters’ afterwards.

Save Parameters: Save the parameters calculated from the dark calibration.

Change Offset: Can be used to change the offset of the device to ensure that background noise is small. Is automatically saved.

Change Gain: Can be used to change the gain of the device to ensure that the measurement range is used optimally. Is automatically saved.

1.3.4.3 Additional Values

California Title 24 JA 10

Opens panel for California Title 24 JA 10. The panels allows you to make a California Title 24 measurement and guides you through the process (3 separate measurements for 100% light, 20% light and minimum light).

1.3.4.4 Info

About

Displays an ‘about’ panel, which shows current software version, device ID and other information.

2 Firmware Documentation

2.1 Flicker Firmware Commands

2.1.1 Introduction

Firmware commands can be sent over the USB port, to communicate with the device directly. They need to end with a carriage return \r.
This interface is designed as a virtual COM port, so it can be handled similarly to a serial port with the settings 8n1/ no protocol. The baudrate must be set to 115200, which is only symbolic.

2.1.2 Overview of all Commands

This overview can be listed by sending the *help? command to the JETI ACC Flicker device.

Debug

*IDN?	-> get device ID
*TRI:IDN?	-> get device ID
*VERSion?	-> get firmware version
*TRI:VERSion?	-> get firmware version
*RST	-> device software reset
*TRI:RST	-> device software reset
*BOOT	-> jump to boot loader
*TRI:BOOT	-> jump to boot loader
*DEBUG	-> enter firmware debug mode
*TRI:DEBUG	-> enter firmware debug mode
*TRI:CONFigure:DEVTYPE	-> set device type (0 – Flicker Display, 1 – Flicker Lamp)
*TRI:CONFigure:DEVTYPE?	-> get device type
*HELP?	-> help text for all commands

Parameters

*TRI:PARAmeter:DEVNUMber	-> set device serial number
*TRI:PARAmeter:DEVNUMber?	-> get device serial number
*TRI:PARAmeter:ADCResolution	-> set ADC resolution (8...16 [bit])
*TRI:PARAmeter:ADCResolution?	-> get ADC resolution [bit]
*TRI:PARAmeter:ADCVoltage	-> set ADC full scale value (2 or 4[V])
*TRI:PARAmeter:ADCVoltage?	-> get ADC full scale value [V]
*TRI:PARAmeter:FASTscan	-> set time to next fastscan cycle (0...300 [ms])
*TRI:PARAmeter:FASTscan?	-> get time to next fastscan cycle in [ms]
*TRI:PARAmeter:TEMPOffs	-> set Temperature offset value (-20.0...+20.0 [K])
*TRI:PARAmeter:TEMPOffs?	-> get Temperature offset value [K]
*TRI:PARAmeter:SAVE	-> save parameters to internal flash storage

*TRI:PARAmeter:ALLPARAmeter?	-> list of all device parameters
*TRI:PARAmeter:OFFSet	-> set offset value (-300...+300 [mV])
*TRI:PARAmeter:OFFSet?	-> get offset value [mV]
*TRI:PARAmeter:GAIN	-> set gain value (1.0...5.0)
*TRI:PARAmeter:GAIN?	-> get gain value
*TRI:PARAmeter:ADAPTGAIN	-> set adaption gain value
*TRI:PARAmeter:ADAPTGAIN?	-> get adaption gain value

Measurement

*TRI:FLICMEASure	-> perform flicker measurement
*TRI:PSTLM	-> perform PstLM measure
*TRI:TEMPErature	-> perform temperature measurement
*TRI:DRKCAL	-> perform flicker dark calibration measurements
*TRI:DRKCAL?	-> get flicker dark calibration values

Read out

*TRI:FETCH:SAMPle	-> fetch sample data (8192Hz), after FLICMEASure
*TRI:FETCH:RAWSAMPle	-> fetch raw sample data (65kHz), after FLICMEASure
*TRI:FETCH:POWERspec	-> fetch power spectrum, after FLICMEASure
*TRI:FETCH:PEAKSPECtrum	-> fetch peak spectrum, after FLICMEASure
*TRI:FETCH:PERCent	-> fetch flicker percent, after FLICMEASure
*TRI:FETCH:INDEX	-> fetch flicker index, after FLICMEASure
*TRI:FETCH:RIPPLE	-> fetch luminous ripple, after FLICMEASure
*TRI:FETCH:CONTRASTflicker	-> fetch contrast flicker, after FLICMEASure
*TRI:FETCH:JEITA	-> fetch JEITA level, after FLICMEASure
*TRI:FETCH:JPOWER	-> fetch JEITA Power Spectrum, after FLICMEASure
*TRI:FETCH:PEAKFREQuency	-> fetch peak frequency, after FLICMEASure
*TRI:FETCH:TINT	-> get last used adaption gain
*TRI:FETCH:LEVEL	-> max of the measured spectrum, after FLICMEASure
*TRI:STATus:ERRor?	-> get the error code
*TRI:STATus:TXTERRor?	-> get error code and description of the error

**TRI:IDN? / *IDN?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns message JETI_TRIFLIC_DISPLAY or JETI_TRIFLIC_LAMP, the current device ID.

Arguments:
none

Example:
*TRI:IDN?<CR>

Response:
JETI_TRIFLIC_DISPLAY<CR>
JETI_TRIFLIC_LAMP<CR>

Depending on device, it returns JETI_TRIFLIC_DISPLAY or JETI_TRIFLIC_LAMP

**TRI:VERS? / *VERS?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the firmware version string.

Arguments:
none

Example:
*TRI:VERS?<CR>

Response:
TRIFIRM VERSION 1.6.1 190324<CR>

**TRI:RST? / *RST?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Performs a device software reset.

Arguments:
none

Example:
*TRI:RST?<CR>

Response:
Performing software reset…<CR>

**TRI:BOOT? / *BOOT?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Jump to bootloader.

Arguments:
none

Example:
*TRI:BOOT?<CR>

Response:
<ACK>

**TRI:DEBUG <mode>/ *DEBUG <mode>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Switches Debug mode on and off (displays additional information).

Arguments:
<mode> 0 – Debug mode off
1 – Debug mode on (additional information)
2 – Debug mode on (more additional information)

Example:
*TRI:DEBUG 1<CR>

Response:
<ACK> or <NAK>

**TRI:CONFigure:DEVTYPE <type>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Sets the device type (display, lamp or tricolor measurement)

Arguments:
<type> 0 – ACC Flicker Display
1 – ACC Flicker Lamp
2 – TriCol

Example:
*TRI:CONF:DEVTYPE 0<CR>

Response:
<ACK> or <NAK>

**TRI:CONFigure:DEVTYPE?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the device type (Display, lamp or tricolor measurement)

Arguments:
none

Example:
*TRI:CONF:DEVTYPE?<CR>

Response:
0 (flicker display)<CR>

**HELP?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Displays command list and a short explanation (The list shown above).

Arguments:
none

Example:
*HELP?<CR>

Response:
Displays a list of possible commands and a short explanation

**TRI:PARAmeter:DEVNUMber <string>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Sets the device serial number.

Arguments:
<string> string with up to 15 character (0…9, a…z)

Example:
*TRI:PARA:DEVNUM 123456abc<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:DEVNUMber?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the device serial number.

Arguments:
none

Example:
*TRI:PARA:DEVNUM?<CR>

Response:
123456abc<CR>

**TRI:PARAmeter:ADCResolution <res>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Sets the ADC resolution for converting the signal from the photo diodes.

Arguments:
<res> - ADC resolution in [bits]
- valid range 8…16
- default: 16

Example:
*TRI:PARA:ADCR 16<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:ADCResolution?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the ADC resolution.

Arguments:
none

Example:
*TRI:PARA:ADCR?<CR>

Response:
16<CR>

**TRI:PARAmeter:ADCVoltage <volt>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Sets the ADC input voltage range.

Arguments:
<volt> - ADC input voltage in [V]
- valid values are 2 and 4
- default: 2

Example:
*TRI:PARA:ADCV 4<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:ADCVoltage?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the ADC input voltage range.

Arguments:
none

Example:
*TRI:PARA:ADCV?<CR>

Response:
4 V<CR>

**TRI:PARAmeter:FASTscan <interval>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Set the time interval for internal fast-scan.
A fast-scan performs a fast readout of the photo diodes if the device is in idle mode.

Arguments:
<interval> - time interval in [ms]
- valid range 1…300 ms
- 0 – no fast-scan will be performed
- default: 0

Example:
*TRI:PARA:FAST 50<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:FASTscan?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the time interval for internal fast-scan.

Arguments:
none

Example:
*TRI:PARA:FAST?<CR>

Response:
50 ms<CR>

**TRI:PARAmeter:TEMPOffs <offset>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Set the temperature offset value.

Arguments:
<offset> - temperature offset in [K]
- valid range -20.00…+20.00
- default: 0.0

Example:
*TRI:PARA:TEMPO 5.25<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:TEMPOffs?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the temperature offset value.

Arguments:
none

Example:
*TRI:PARA:TEMPO?<CR>

Response:
5.25 K<CR>

**TRI:PARAmeter:OFFSet <offset>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Set the ADC offset value.

Arguments:
<offset> - ADC offset in [mV]
- valid range -300…+300
- default: 0

Example:
*TRI:PARA:OFFS 150<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:OFFSet?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the ADC offset.

Arguments:
none

Example:
*TRI:PARA:OFFS?<CR>

Response:
150 mV<CR>

**TRI:PARAmeter:GAIN <gain>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Set the ADC gain value.

Arguments:

<gain>	- ADC gain - valid range 1.0…5.0 - default: 1.0

Example:
*TRI:PARA:GAIN 1.5<CR>

Response:
<ACK> or <NAK>

**TRI:PARAmeter:GAIN?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Queries the ADC gain value.

Arguments:
none

Example:
*TRI:PARA:GAIN?<CR>

Response:
1.5<CR>

**TRI:PARAmeter:ALLPARAmeter?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns a list of all parameter settings.

Arguments:
none

Example:
*TRI:PARA:ALLPARA?<CR>

Response:
List of firmware parameter settings

**TRI:PARAmeter:SAVE*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Write parameters to internal flash storage.

Arguments:
none

Example:
*TRI:PARA:SAVE<CR>

Response:
<ACK> or <NAK>

**TRI:FLICMEASure <gain>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Performs a flicker measurement.
The internal sampling rate is about 1.25MHz. 19-fold oversampling results in an effective sampling rate of approx. 65KHz (see *TRI:FETCH:RAWSAMPle). The signal is then filtered again, resulting in a sample signal of 8KHz (see *TRI:FETCH:SAMPle).
Depending on the device type several flicker values are calculated.
To get the calculated frequency, flicker percentage, flicker index, luminous ripple, contrast flicker and JEITA value use the appropriate fetch commands.

Arguments:

<gain> - gain
- possible values from 0 to 3 (0 means adaption mode)
- default: 0

Example:
*TRI:FLICMEAS 0<CR>

Response:
<ACK><BEL> or <NAK>

**TRI:PSTLM <tint> <duration>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Performs a PstLM measurement.

Arguments:
<tint> - integration time in µs, possible values from 1 to 100
<duration> - measurement duration in s, possible values from 1 to 600

Example:
*TRI:PSTLM 10 600<CR>

Response:
<ACK><PSTLM data> or <NAK>

**TRI:TEMPE*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Perform an ambient temperature measurement.

Arguments:
None

Example:
*TRI:TEMPE<CR>

Response:
24.57 °C<CR>

**TRI:DRKCAL*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Perform flicker dark calibration measurements. This will take a while, until temperature stabilizes.

Arguments:
none

Example:
*TRI:DRKCAL<CR>

Response:
<ACK><BEL> or <NAK>

**TRI:DRKCAL?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Get the determined flicker dark calibration values.

Arguments:
none

Example:
*TRI:DRKCAL?<CR>

Response:
The determined dark calibration values.

**TRI:FETCH:SAMPle <format>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

After a *TRI:FLICMEASure command, *TRI:FETCH:SAMPle can be used to retrieve the acquired sample data.
Using *TRI:FETCH:SAMPle, a 1/2 s sample with 8 kHz can be obtained. This corresponds to 4096 values (raw counts).

Arguments:

<format> - 1 – binary output
- 2 – ASCII output
- default: 1

Example:
*TRI:FETCH:SAMP<CR>

Response:
If argument was 1, binary data is sent, which is structured as 16-bit unsigned short integer. The first two bytes are a header, which gives the length of the data sent as 16-bit unsigned short integer. Right now, it is 8192 bytes (4096 measurement values as 16-bit unsigned short int). Following that, the actual data begins.

If argument was 2, ASCII data is sent. Acquired measurement numbers are displayed as characters, each ending with a carriage return. The last line terminated with a carriage return is followed by an end-of-text byte (ETX) to signal the end.

**TRI:FETCH:RAWSAMPle <format>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

After a *TRI:FLICMEASure command, *TRI:FETCH:RAWSAMPle can be used to retrieve the acquired raw unfiltered sample data.
Using *TRI:FETCH:RAWSAMPle, an unfiltered 65 kHz sample of 1/8 s length can be obtained. This corresponds to 8192 values (raw counts).

Arguments:

<format> - 1 – binary output
- 2 – ASCII output
- default: 1

Example:
*TRI:FETCH:RAWSAMP<CR>

Response:
If argument was 1, binary data is sent, which is structured as 16-bit unsigned short integer. The first two bytes are a header, which gives the length of the data sent as 16-bit unsigned short integer. Right now, it is 16384 bytes (8192 measurement values as 16-bit unsigned short int). Following that, the actual data begins.

If argument was 2, ASCII data is sent. Acquired measurement numbers are displayed as characters, each ending with a carriage return. The last line terminated with a carriage return is followed by an end-of-text byte (ETX) to signal the end.

**TRI:FETCH:POWERspec <format>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Power spectrum data is sent. This can be used after a *TRI:FLICMEASure command to retrieve the power spectrum of the acquired sample.
The power spectrum results from a Fourier transform of the sample data. Since there are 4096 values for 0.5 s, this results in 2048 frequency values in 2 Hz steps.

Arguments:

<format> - 1 – binary output
- 2 – ASCII output
- default: 1

Example:
*TRI:FETCH:POWER<CR>

Response:
If argument was 1, binary data is sent, which is structured as 32-bit float. The first two bytes are a header, which gives the length of the data sent as 16-bit unsigned short integer. Right now, it is 8192 bytes (2048 power spectrum values as 32-bit float). Following are the actual power spectrum values.

If argument was 2, ASCII data is sent. Power spectrum values are displayed as characters, each ending with a carriage return. The last line terminated with a carriage return is followed by an end-of-text byte (ETX) to signal the end.

**TRI:FETCH:PEAKSPECtrum <format>*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Peak spectrum data is sent. This can be used after a *TRI:FLICMEASure command to retrieve the peak spectrum of the acquired sample.
The ‘peak spectrum’ is just the power spectrum with the DC value removed.

Arguments:

<format>	- 1 – binary output - 2 – ASCII output - default: 1

Example:
*TRI:FETCH:PEAKSPEC<CR>

Response:
If argument was 1, binary data is sent, which is structured as 16-bit unsigned short integer. The first two bytes are a header, which gives the length of the data sent as 16-bit unsigned short integer. Right now, it is 4096 bytes (2048 peak spectrum values as 16-bit unsigned short integer). Following are the actual peak spectrum values.

If argument was 2, ASCII data is sent. Peak spectrum values are displayed as characters, each ending with a carriage return. The last line terminated with a carriage return is followed by an end-of-text byte (ETX) to signal the end.

**TRI:FETCH:PERCent*

☐ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns the calculated flicker percentage.

Arguments:
none

Example:
*TRI:FETCH:PERC<CR>

Response:
Returns percent flicker rounded to 1 decimal place.

**TRI:FETCH:RIPPLE*

☐ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns the calculated luminous ripple value.

Arguments:
none

Example:
*TRI:FETCH:RIPPLE<CR>

Response:
Returns luminous ripple rounded to 1 decimal place.

**TRI:FETCH:CONTRASTflicker*

☒ *ACC Flicker Display* ☐ *ACC Flicker Lamp*

Returns the calculated flicker contrast value.

Arguments:
none

Example:
*TRI:FETCH:CONTRAST<CR>

Response:
Returns flicker contrast rounded to 2 decimal place.

**TRI:FETCH:INDEX*

☐ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns the calculated flicker index

Arguments:
none

Example:
*TRI:FETCH:INDEX<CR>

Response:
Returns flicker index rounded to 2 decimal places.

**TRI:FETCH:JEITA*

☒ *ACC Flicker Display* ☐ *ACC Flicker Lamp*

Returns the calculated JEITA value.

Arguments:
none

Example:
*TRI:FETCH:JEITA<CR>

Response:
Returns JEITA value.

**TRI:FETCH:JPOWER <format>*

☒ *ACC Flicker Display* ☐ *ACC Flicker Lamp*

JEITA power spectrum data is sent. This can be used after a *TRI:FLICMEASure command to retrieve the JEITA power spectrum of the acquired sample.

Arguments:

<format>	- 1 – binary output - 2 – ASCII output - default: 1

Example:
*TRI:FETCH:JPOWER<CR>

Response:
If argument was 1, binary data is sent, which is structured as 32-bit float. The first two bytes are a header, which gives the length of the data sent as 16-bit unsigned short integer. Right now, it is 400 bytes (100 32-bit float or power spectrum values). Following are the actual power spectrum values.

If argument was 2, ASCII data is sent. Power spectrum values are displayed as characters, each ending with a carriage return. The last line terminated with a carriage return is followed by an end-of-text byte (ETX) to signal the end.

**TRI:FETCH:PEAKFREQuency*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Returns the frequency of the highest peak (unless there are higher multiples of the peak, e. g. 300 Hz and 600 Hz, in which case the lower frequency is chosen).

Arguments:
none

Example:
*TRI:FETCH:PEAKFREQ<CR>

Response:
Returns the frequency of the highest peak, rounded to full hertz.

**TRI:FETCH:TINT*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Get the last used integration gain.

Arguments:
None

Example:
*TRI:FETCH:TINT<CR>

Response:
Returns the last used integration gain.

**TRI:FETCH:LEVEL*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Get the maximum of the previously measured spectrum in counts and %.

Arguments:
None

Example:
*TRI:FETCH:LEVEL<CR>

Response:
Returns the maximum of the last measured spectrum in counts and %.

**TRI:STATus:ERRor?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Displays the error code.

Arguments:
None

Example:
*TRI:STATus:ERRor?<CR>

Response:
Displays the error code.

**TRI:STATus:TXTERRor?*

☒ *ACC Flicker Display* ☒ *ACC Flicker Lamp*

Get the error code and a description of the error.

Arguments:
None

Example:
*TRI:STATus:ERRor?<CR>

Response:
Displays the error code and a description.